UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
INTRODUÇÃO À MECÂNICA DOS FLUÍDOS

Simulação – Calculo de pressão e velocidade em uma cavidade utilizando o método das diferenças finitas

ANTONIO JONAS GONÇALVES - 2016021023

THIAGO ALVES DE ARAUJO - 2016019787

JOÃO PESSOA, 2020

UFPB

Resumo: O projeto tem como objetivo realizar a simulação do *problema da cavidade (Cavity flow).* Este problema consiste em calcular a pressão e a velocidade em uma cavidade usando o método das diferenças finitas para resolver as equações de Navier-Stokes. Para tal simulação, foi utilizado a linguagem de programação python.

JOÃO PESSOA, 2020

1 – Introdução

Problemas de mecânica dos fluidos estão sempre presentes em nosso cotidiano, especialmente em problemas de engenharia. Para soluciona-los, podemos utilizar ferramentas computacionais que, através de modelagem física e matemática, nos entrega resultados muito satisfatórios, com tempo de execução e custos de desenvolvimento muito menores quando comparado com métodos experimentais.

Neste trabalho, vamos realizar a resolução de um escoamento bidimensional no interior de uma cavidade. Para isso, utilizaremos as equações de Navier-Stokes (2D) para velocidade e a equação de Poisson para pressão como podemos observar abaixo.

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$$

Figura 1 Equações de NS2D

$$\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}=-\rho\left(\frac{\partial u}{\partial x}\frac{\partial u}{\partial x}+2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+\frac{\partial v}{\partial y}\frac{\partial v}{\partial y}\right)$$

Figura 2 Equação de Poisson

2 – Esquema matemático

Iniciamos nossa modelagem calculando o campo de pressão, assumindo que:

$$\nabla \mathbf{u}^{n+1/2} \neq 0.$$

Assim, temos:

$$\nabla^2 p^{n+1}=\frac{\rho}{\Delta t}\nabla\cdot\mathbf{u}^{n+1/2}-\rho\left(\left(\frac{\partial u}{\partial x}\right)^2+2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+\left(\frac{\partial u}{\partial y}\right)^2\right)$$

Agora, vamos discretizar as equações apresentadas anteriormente. Para a equação de velocidade em ***u*** e ***v***, temos:

$$\frac{u_{i,j}^{n+1}-u_{i,j}^{n}}{\Delta t}+u_{i,j}^{n}\frac{u_{i,j}^{n}-u_{i-1,j}^{n}}{\Delta x}+v_{i,j}^{n}\frac{u_{i,j}^{n}-u_{i,j-1}^{n}}{\Delta y}=$$
$$-\frac{1}{\rho}\frac{p_{i+1,j}^{n}-p_{i-1,j}^{n}}{2\Delta x}$$
$$+\nu\left(\frac{u_{i+1,j}^{n}-2u_{i,j}^{n}+u_{i-1,j}^{n}}{\Delta x^2}+\frac{u_{i,j+1}^{n}-2u_{i,j}^{n}+u_{i,j-1}^{n}}{\Delta y^2}\right)$$

*Figura 3 Equação de momento para **u***

$$\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta t} + u_{i,j}^n \frac{v_{i,j}^n - v_{i-1,j}^n}{\Delta x} + v_{i,j}^n \frac{v_{i,j}^n - v_{i,j-1}^n}{\Delta y} =$$
$$-\frac{1}{\rho}\frac{p_{i,j+1}^n - p_{i,j-1}^n}{2\Delta y}$$
$$+\nu\left(\frac{v_{i+1,j}^n - 2v_{i,j}^n + v_{i-1,j}^n}{\Delta x^2} + \frac{v_{i,j+1}^n - 2v_{i,j}^n + v_{i,j-1}^n}{\Delta y^2}\right)$$

*Figura 4 Equação de momento para **v***

Para a equação de pressão obtemos a seguinte expressão:

$$\frac{p_{i+1,j}^n - 2p_{i,j}^n + p_{i-1,j}^n}{\Delta x^2} + \frac{p_{i,j+1}^n - 2p_{i,j}^n + p_{i,j-1}^n}{\Delta y^2} =$$
$$\rho\left[\frac{1}{\Delta t}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\right)\right.$$
$$-\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - 2\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x}$$
$$\left.-\frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\right]$$

Figura 5 Equação de Poisson para pressão, discretizada.

Então, vamos definir as condições de limite de pressão. Iniciamos escrevendo a equação de Navier-Stokes da seguinte forma:

$$\nabla p = \rho\left(-\frac{\partial \mathbf{u}}{\partial t} - \mathbf{u}\cdot\nabla\mathbf{u} + \nu\nabla^2\mathbf{u}\right)$$

Multiplicamos em ambos os lados com o vetor normal do limite **n** e obtemos a seguinte equação para a pressão:

$$\mathbf{n}\cdot\nabla p = \rho\mathbf{n}\cdot\left(-\frac{\partial \mathbf{u}}{\partial t} - \mathbf{u}\cdot\nabla\mathbf{u} + \nu\nabla^2\mathbf{u}\right)$$

Em uma fronteira rígida, $\mathbf{u_1}$ e $\mathbf{u_2}$ são 0, simplificando a equação anterior. Com a condição de limite para uma fronteira rígida com y = 0, obtemos:

$$\frac{\partial p}{\partial y} = \rho\nu\frac{\partial^2 u_2}{\partial x_2^2}$$

Por fim, vamos escrever as equações de momento para a velocidade em **u** e **v**. Para direção **u** temos:

$$u_{i,j}^{n+1} = u_{i,j}^n - u_{i,j}^n\frac{\Delta t}{\Delta x}(u_{i,j}^n - u_{i-1,j}^n) - v_{i,j}^n\frac{\Delta t}{\Delta y}(u_{i,j}^n - u_{i,j-1}^n)$$
$$-\frac{\Delta t}{\rho 2\Delta x}(p_{i+1,j}^n - p_{i-1,j}^n) + \nu\left(\frac{\Delta t}{\Delta x^2}(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + \frac{\Delta t}{\Delta y^2}(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n)\right)$$

Para a direção **v** temos:

$$v_{i,j}^{n+1} = v_{i,j}^{n} - u_{i,j}^{n} \frac{\Delta t}{\Delta x} (v_{i,j}^{n} - v_{i-1,j}^{n}) - v_{i,j}^{n} \frac{\Delta t}{\Delta y} (v_{i,j}^{n} - v_{i,j-1}^{n})$$

$$- \frac{\Delta t}{\rho 2 \Delta y} (p_{i,j+1}^{n} - p_{i,j-1}^{n}) + \nu \left(\frac{\Delta t}{\Delta x^2} (v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}) + \frac{\Delta t}{\Delta y^2} (v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n}) \right)$$

3 – Código

Para melhor entendimento do código, ele foi separado em duas funções principais: A primeira é responsável por realizar o cálculo da pressão, enquanto a segunda possui os cálculos da velocidade. Abaixo podemos ver os trechos do código.

Função do calculo da pressão

```
In [2]: def presPoisson(p, dx, dy,rho,nu,u,v):
            pn = np.empty_like(p)
            pn = p.copy()

            #Term in square brackets
            b[1:-1,1:-1]=rho*(1/dt*((u[1:-1,2:]-u[1:-1,0:-2])/(2*dx)+(v[2:,1:-1]-v[0:-2,1:-1])/(2*dy))-\
                         ((u[1:-1,2:]-u[1:-1,0:-2])/(2*dx))**2-\
                         2*((u[2:,1:-1]-u[0:-2,1:-1])/(2*dy)*(v[1:-1,2:]-v[1:-1,0:-2])/(2*dx))-\
                         ((v[2:,1:-1]-v[0:-2,1:-1])/(2*dy))**2)

            for q in range(nit):
                pn = p.copy()
                p[1:-1,1:-1] = ((pn[1:-1,2:]+pn[1:-1,0:-2])*dy**2+(pn[2:,1:-1]+pn[0:-2,1:-1])*dx**2)/\
                                (2*(dx**2+dy**2)) -\
                                dx**2*dy**2/(2*(dx**2+dy**2))*b[1:-1,1:-1]

                ##at y = 2
                p[-1,:] = p[-2,:] - rho*nu/dy*(-2*v[-2,:]+v[-3,:])

                #at y = 0
                p[0,:] = p[1,:] - rho*nu/dy*(-2*v[1,:]+v[2,:])

                #at x = 0
                p[:,0] = p[:,1] - rho*nu/dx*(-2*u[:,1]+u[:,2])

                #at x = 2
                p[:,-1] = p[:,-2] - rho*nu/dx*(-2*u[:,-2]+u[:,-3])

            return p
```

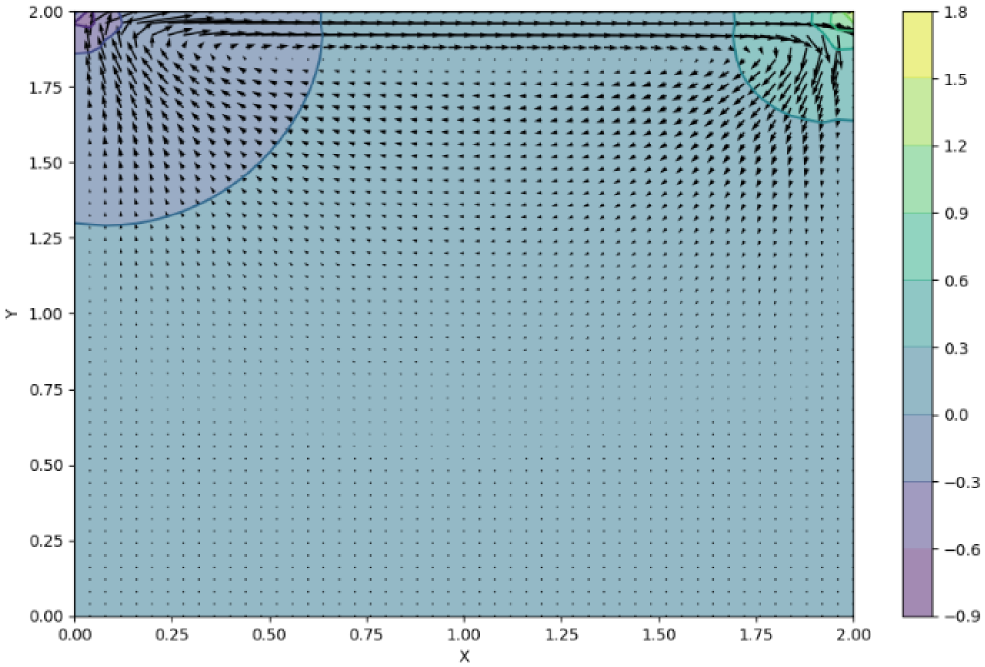
Figura 6 Função "pressão"

```
In [3]: def cavityFlow(nt, u, v, dt, dx, dy, p, rho, nu):
            un = np.empty_like(u)
            vn = np.empty_like(v)


            for n in range(nt):
                un = u.copy()
                vn = v.copy()

                p = presPoisson(p, dx, dy, rho, nu, u, v)

                u[1:-1,1:-1] = un[1:-1,1:-1]-\
                                un[1:-1,1:-1]*dt/dx*(un[1:-1,1:-1]-un[1:-1,0:-2])-\
                                vn[1:-1,1:-1]*dt/dy*(un[1:-1,1:-1]-un[0:-2,1:-1])-\
                                dt/(2*rho*dx)*(p[1:-1,2:]-p[1:-1,0:-2])+\
                                nu*(dt/dx**2*(un[1:-1,2:]-2*un[1:-1,1:-1]+un[1:-1,0:-2])+\
                                dt/dy**2*(un[2:,1:-1]-2*un[1:-1,1:-1]+un[0:-2,1:-1]))

                v[1:-1,1:-1] = vn[1:-1,1:-1]-\
                                un[1:-1,1:-1]*dt/dx*(vn[1:-1,1:-1]-vn[1:-1,0:-2])-\
                                vn[1:-1,1:-1]*dt/dy*(vn[1:-1,1:-1]-vn[0:-2,1:-1])-\
                                dt/(2*rho*dy)*(p[2:,1:-1]-p[0:-2,1:-1])+\
                                nu*(dt/dx**2*(vn[1:-1,2:]-2*vn[1:-1,1:-1]+vn[1:-1,0:-2])+\
                                (dt/dy**2*(vn[2:,1:-1]-2*vn[1:-1,1:-1]+vn[0:-2,1:-1])))

                u[0,:] = 0
                u[:,0] = 0
                u[:,-1] = 0
                u[-1,:] = Uwall     #set velocity on cavity lid equal to Uwall
                v[0,:] = 0
                v[-1,:]=0
                v[:,0] = 0
                v[:,-1] = 0


            return u, v, p
```

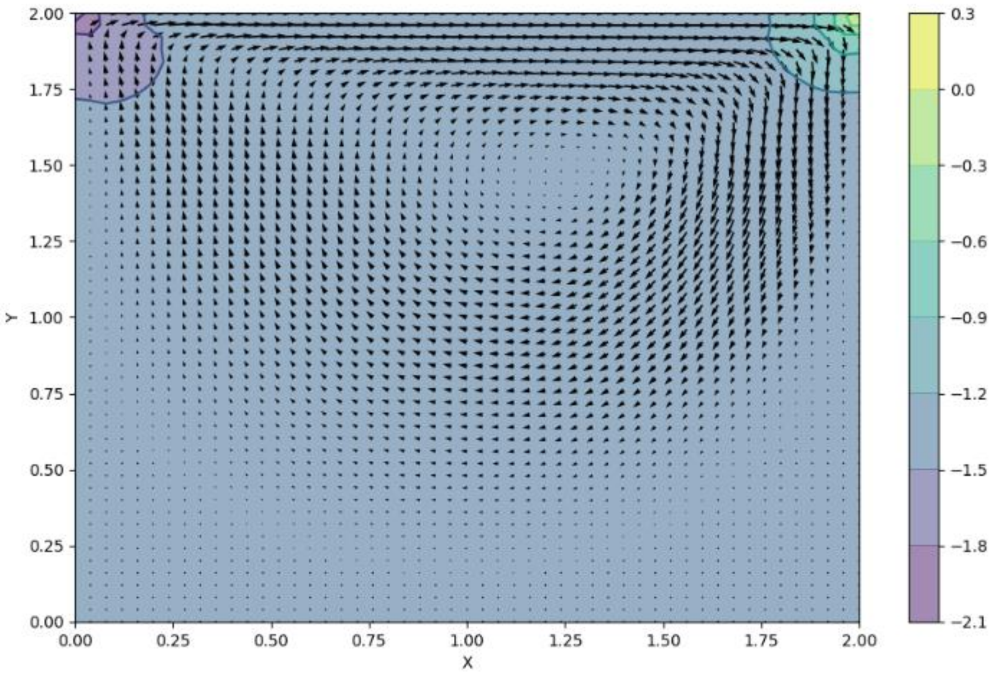
Figura 7 Função "Cavity Flow"

4 – Resultados

Vamos iniciar a execução com um baixo tempo de simulação. Abaixo podemos observar o resultado obtido com nt = 100.

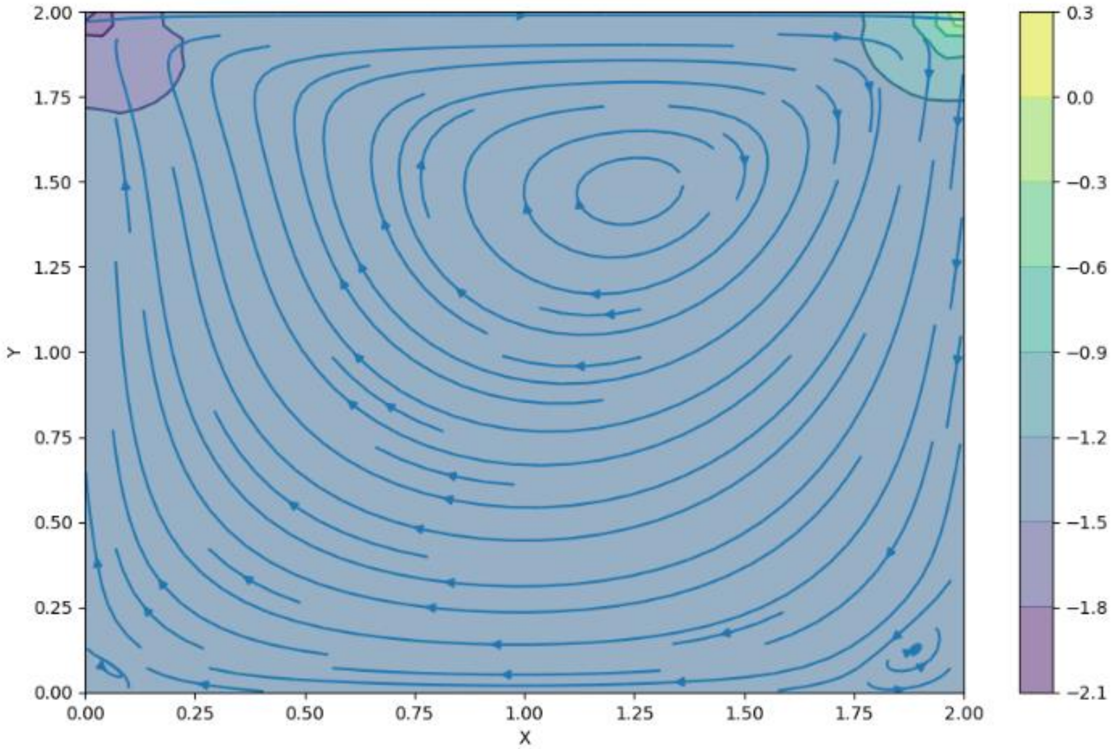


Como podemos observar, duas zonas de alta pressão estão se formando nas extremidades (A escala ao lado representa a pressão em Pa). Assim, vamos aumentar o valor de nt para observar quanto tempo o sistema leva para estabilizar. Com nt = 5000 temos:



Como podemos observar, as zonas de pressão diminuíram substancialmente (observe os valores na escala da pressão).

Por fim, podemos utilizar um diagrama de fluxo para observar o fluxo na cavidade. Abaixo podemos ver o resultado obtido.



5 – Conclusão

Como podemos observar, a modelagem matemática juntamente com as ferramentas computacionais se mostrou eficaz para a simulação do problema da cavidade. Além disso, a modelagem desenvolvida pode ser facilmente adaptada de acordo com a situação desejada, bastando apenas o usuário regular os parâmetros de entrada, tornando a simulação muito mais eficaz do que uma modelagem experimental.